FrameworkComparisons

Some of this information can be found elsewhere in the Tapestry documentation.

Reasons for using Tapestry

A colleague of mine recently reminded me that in order to ***really*** know you want to use something, you have to know when not to use it.

When to use Tapestry

- If you have a one or more designers (the artsy-people) who like to modify HTML documents instead of things like JSPs.
- If you don't like dealing with XML configuration or "boiler plate" code (Tapestry 5)
- If you have struggled in the past with separating Model from the View (MVC aka Model 2 design pattern), Tapestry makes this much easier.
- If you want a component based framework based on POJOs (Plain Old Java Objects).
- If you need high performance and high load handling.

When not to use Tapestry

- If you are stuck on Java 1.4 or are otherwise forbidden from using annotations and Java 1.5 features.
- If you require out-of-the-box Portlet support (check for updates, or re-word this, may be available in T4/T5)
- If you are afraid or forbidden to learn new technology/frameworks/patterns (IOC, Maven, etc)
- If you require first class IDE support. There have been IDE plugins in the past (T4), but we must wait a while longer for T5 IDE plugins (update).

Challenges faced by Tapestry newcomers

Some of these are addressed elsewhere in the Tapestry documentation.

- Static Structure, Dynamic Behavior. This is not as limiting as it may seem, and the benefits are worth it. Usually this issue comes up when users want hot-pluggable components, like what you can get using a PHP based Content Management System. In almost all cases, the user just needs to apply some creative thinking in order to accomplish the goal, while still using a static structure. It may also require you to redefine your notion of "structure" and "behavior" depending on which frameworks you have used previously.
- Using Maven. For many users, Tapestry 5 may introduce you to Maven for the first time. Many T5 tutorials use Maven to get you started (archetypes, screencasts, articles, appfuse), but you are not required to use Maven at all. To restate, Maven has never been a requirement, only a convenient way to deal with dependencies and builds. So it may be a development burden to stay up to date if your particular build tool doesn't find that dependency of a dependency of a dependency for you.

Quick Comparisons to Other Frameworks

- Tapestry And Struts
  - See TapestryFasttrackForStrutsProgrammers
- Tapestry And Wicket
  - 2007: See author Kent Tong's blog post "My Thoughts On The Differences" http://agileskills2.org/blog/2007/09/my_thoughts_on_the_differences.html
- Tapestry And JSF
  - placeholder
- Tapestry And Flex
  - placeholder
- Tapestry And GWT (Google Web Toolkit)
  - placeholder
- Comparisons of Multiple Frameworks, including Tapestry
  - 2010: Matt Raible at Devoxx 2010
  - 2010: Java EE Productivity Report 2011 survey by ZeroTurnaround